Quantum Leap and Crash: Searching and Finding Bias in Quantum Random Number Generators

DARREN HURLEY-SMITH*, Royal Holloway University of London
JULIO HERNANDEZ-CASTRO†, University of Kent

Random numbers are essential for cryptography and scientific simulation. Generating truly random numbers for cryptography can be a slow and expensive process. Quantum physics offers a variety of promising solutions to this challenge, proposing sources of entropy that may be genuinely unpredictable, based on the inherent randomness of certain physical phenomena. These properties have been employed to design Quantum Random Number Generators (QRNGs), some of which are commercially available. In this work, we present the first published analysis of the Quantis family of QRNGs (excluding AIS-31 models), designed and manufactured by ID Quantique (IDQ). Our study also includes Comscire's PQ32MU QRNG, and two online services: the Australian National University's (ANU) QRNG, and the Humboldt Physik generator.

Each QRNG is analysed using 5 batteries of statistical tests: Dieharder, National Institute of Standards and Technology (NIST) SP800-22, Ent, Tuftests and TestU01, as part of our thorough examination of their output. Our analysis highlights issues with current certification schemes, which largely rely on NIST SP800-22 and Diehard tests of randomness. We find that more recent tests of randomness identify issues in the output of QRNG, highlighting the need for mandatory post-processing even for low-security usage of random numbers sourced from QRNGs.

CCS Concepts: • **Security and privacy → Security requirements**; *Mathematical foundations of cryptography*.

Additional Key Words and Phrases: quantum random number generation, entropy, cryptography, statistical analysis

ACM Reference Format:
Darren Hurley-Smith and Julio Hernandez-Castro. 2019. *Quantum Leap and Crash:* Searching and Finding Bias in Quantum Random Number Generators. *ACM Trans. Priv. Sec.* ?, ?, Article ? (? 2019), 25 pages. https://doi.org/?

1 INTRODUCTION

Providing Pseudo-Random Number Generator (PRNG) algorithms with entropy is a problem approached in many ways. Harvesting it from human-interface devices is one method, but this depends on a steady stream of actions from a user. However, devices harnessing naturally unpredictable phenomena have been developed, which can be used to seed PRNGs or act as RNGs in their own right.

True Random Number Generators (TRNG) harvest entropy from a variety of hardware-specific phenomena (such as leakage in ring-oscillators), process the output and forward it to a host device. TRNGs use a variety of classical noise sources for their entropy.

Quantum random number generators (QRNG) harness the inherently unpredictable properties of quantum phenomena to provide entropy. Rarity et al. [1] discussed the possibility of combining quantum random number generation and key sharing. Stefanov et al. [2] outlined the scientific

Authors' addresses: Darren Hurley-Smith, darren.hurley-smith@rhul.ac.uk, Royal Holloway University of London, Egham Hil, Egham, Surrey, TW20 0EX; Julio Hernandez-Castro, j.c.hernandez-castro@kent.ac.uk, University of Kent, School of Computing, Cornwallis Building, Darwin Road, Canterbury, Kent, CT2 7NZ.

principles and design of an optical quantum random number generator. In their work, they proposed the use of an optical quantum phenomenon in which photons either pass-through or reflect off of a partially-mirrored surface, as a source of entropy. This is the basis of ID Quantique's (IDQ) Quantis brand of QRNGs.

Comscire's PQ32MU uses an alternative entropy source, quantum shot-noise. ANU's QRNG uses broadband measurements of a vacuum field contained in the radio frequency side-bands of a single-mode laser. There is significant diversity in the sources of quantum entropy, which begs the question: are these sources equally unpredictable, and are there different considerations that must be accounted for to correctly sanitise their raw entropy into usable output?

We have observed devices that achieve Common Criteria EAL4+ certification while failing relatively simple statistical tests. It is crucial to identify weak devices to better inform future iterations of testing and best practice in device certification. When manufacturers or service providers claim that their device (or output) is a viable source of randomness for cryptographic operations, one must approach such claims from a position of scepticism. With prices in the multi-thousand dollar range, these devices are targeted towards well-funded customers with critical security requirements. This may lead to significant repercussions should devices exhibit flaws (such as predictable output) despite their certifications.

This paper provides the first in-depth analysis of IDQ's [3] Quantis QRNG 16M PCI-E, 4M PCI-E and USB devices, Comscire's P32MU, the Australian National University's (ANU) QRNG and Humboldt University's Physik Online QRNG service. Dieharder, National Institute of Standards and Technology (NIST) SP800-22, Ent, TestU01 and Tuftests have been used to give a thorough statistical analysis of the output of these devices. Significant biases are identified in the Quantis QRNG modules, and further analysis is conducted in an attempt to characterise these biases. The authors present novel findings highlighting a previously unknown statistical anomaly in the Quantis QRNG which may have a significant impact on specific applications.

Another contribution of this work is to demonstrate that current certification schemes require a wider range of statistical tests to identify non-random output. Mere days of data collection and testing could provide a far more rigorous method of identifying flaws in RNGs that currently go unnoticed. The testing strategies used to certify RNG products, and the need for more stringent tests, have also been discussed. Finally, this paper identifies how user advice and associated manuals must match the current consensus in secure quantum random number generation. Post-processing should never be deemed as optional, as the results for Quantis output in default mode demonstrate.

Previous publication. Preliminary work [4] related to this research has previously been presented at Real World Crypto (RWC) 2018 in Zurich, Switzerland. The work in question has not been published in any conference proceedings or journal. RWC 2018 has no proceedings for any kind. This manuscript greatly expands the early work presented in [4], which presents preliminary findings regarding 3 Quantis RNGs, a ChaosKey and urandom.

Organisation. The rest of this paper is laid out as follows: Section 2 outlines the background of the research, discussing the devices and the theory behind their function. Section 3 discusses the methodology employed in this work. Section 4 shows the results and analyses. Section 5 discusses the implications of the findings presented in this work. Section 6 provides concluding remarks and future work.

2 BACKGROUND

Random number generation is critical for the correct functioning of cryptography and security systems. A recent attack [5] implementing Coppersmith's attack against RFID ID cards in Taiwan

highlighted the importance of robust random number generation. A poorly implemented RNG allowed the factoring of 184 RSA keys out of 2 million Taiwanese ID cards.

Both ID cards were certified as secure at the time of these discoveries, under the FIPS 140-2 standard, and Common Criteria. Researchers commented that the secrecy around implementations of security systems that cannot disclose their designs due to Intellectual Property (IP) considerations. This culture of secrecy makes it all but impossible to independently verify the claims of manufacturers, or verify the outcomes of tests that lead to certification. Furthermore, manufacturers have a vested interest in intercepting or suppressing reverse-engineering attempts that identify flaws, until major attacks make it necessary to disclose vulnerabilities. It is vital that RNGs are robust and unpredictable, but ensuring that this is the case is no easy feat.

QRNGs are frequently described as being attractive for their high speed output. With the PQ32MU providing 32 mega-bits per second (Mb/s) and Quantis 16M producing 16 Mb/s, this is easy to see. But simple USB TRNGs, such as ChaosKey and Bitbabbler are capable of speeds of 8mps and up to 30 Mb/s respectively.

The developers of the Bitbabbler [1] provide a succinct explanation of why the theoretical maximum speed of their devices is not an advisable speed to operate them. As the output rate of the Bitbabbler is increased, the whiteness of the output degrades. This effectively means that the user is trading randomness for speed, and the developers have set a default 2.5 Mb/s for a quad-source device in the software manager. Application-specific Circuits (ASICs) have been used to achieve output speed in excess of 10 giga-bits per second (Gb/s) [6]. However, these are experimental and not yet in commercial circulation.

A survey of TRNGS suitable for implementation in FPGAs [7] was conducted in 2016. The findings of this survey identify a significant increase in design complexity and required area (LuT/LR) to improve the output speed of such devices. Varchola et al. [8] proposed low-power designs (TERO-TRNG) which achieved 1 Mb/s output speeds while consuming only 1.23 mW of energy. Though a remarkable gain in efficiency and speed over previous attempts this design still falls significantly short of speeds achievable by QRNGs that were considered old at the time, such as the Quantis product line. Chercaoui et al. [9] achieve truly impressive speeds of more than 245 Mb/s with a Cyclone V FPGA. However, the ring based oscillator design requires a huge number of cells: 352/256 LuT/LR for a Cyclone V FPGA. Constant verification of events is requires and each cell must be independently initialised at start up. Manual placement of components is required to ensure random output, substantially reducing the convenience and long-term usability of this design. Sivaraman et al. [10] claim to have produced a design capable of over 26 Mb/s using only 8 ring oscillators and 64 inverters in total. They claim that no special configuration is required and output remains characteristic of an independent TRNG regardless of underlying the FPGA. However, this design was only recently (2020) proposed and is unlikely to be implemented for some time. At present, FPGA designs show promise (in terms of both speed and randomness) but lack the commercial infrastructure and battle-tested designs of commercial QRNGs.

Table 1 provides a breakdown of retail cost for a small sample of commercial TRNG (which pass all randomness tests used in this paper) compared with Quantis QRNGs. Speeds were recorded and averaged over 100 x 1 MB streams. Bitbabbler was constrained by a software cap, this can be released and 30 Mb/s achieved, at the cost of reducing the randomness of the output stream. Default configurations and values were used for all devices in the interests of fairness. At present, QRNGs are significantly more expensive than the majority of consumer grade TRNGs (by orders of magnitude), but they produce speeds that classical TRNGs find difficult to achieve without compromising the randomness of their output at this time.

[1] http://www.bitbabbler.org/how.html

Table 1. TRNG and QRNG cost at time of writing

Device	Vendor	Cost	Output Speed (Mb/s)
16M PCI-E	IDQ[2]	$ 3,136	15.8
4M PCI-E	IDQ	$ 1,355	3.98
4M USB	IDQ	$ 1,036	3.92
Chaoskey	vikings[3]	€49	3.95
Bitbabbler Black	Bitbabbler [4]	AUD $ 99	2.49
Bitbabbler White	Bitbabbler	AUD $ 199	2.49

2.1 Quantis QRNGs

ID Quantique (IDQ) [11] were the first to harness quantum phenomena as an entropy source in a commercial product, in 2001[5]. IDQ [3] has commercialised devices that use an optical quantum process as a source of randomness. This allows the generation of random numbers at a speed of 4 Mb/s (4M devices) and 16 Mb/s (16M devices) on PCI-E and USB hardware devices.

A block diagram of the Quantis quantum entropy module can be found in Figure 1. This diagram details the hardware elements of a typical Quantis device. A 4M, whether it connects to a host machine using PCI-E or USB, uses a single Optical entropy source. The 16M uses 4 of these entropy sources, which are used to provide a higher rate of output by simply generating 4 times as many bits to forward through the subsequent elements of the device.

The lowest level of the diagram represents the physical entropy source. A light source is used to generate individual photons, which are directed towards a semi-transparent mirror. There is an approximately 0.5 probability of this mirror emitting or reflecting the photons directed towards it. Two photon detectors are used to track which of these eventualities occurs. If the photon is emitted, a 1 will be output to the acquisition layer. If it is reflected a 0 is output to the acquisition layer.

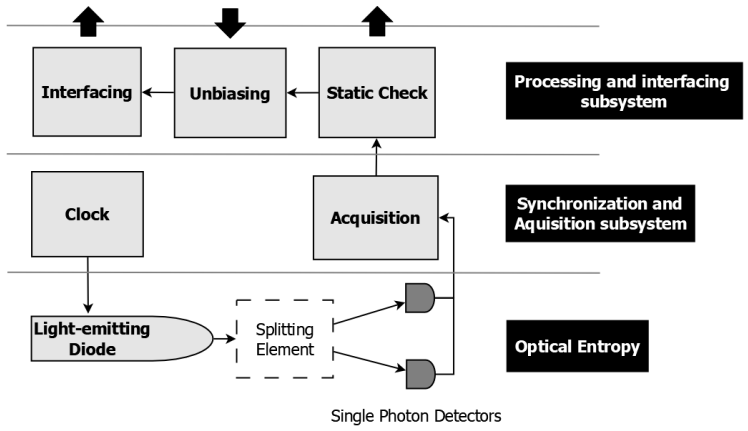


Fig. 1. Quantis QRNG hardware block diagram, adapted from IDQ Quantis 2019 white paper [6]

The synchronisation and acquisition subsystem is responsible for registering the photon detector events and clocking the LED used to generate light pulses. The acquisition subsystem then forwards the bits generated by detector events to the static check subsystem. The static check phase is responsible for identifying any egregious errors, such as the excessive generation of the same bit

[5]http://bit.ly/2Fx3o2C

values. It is intended to provide a means of detecting flaws in the preceding processes and halt the device if such issues are detected.

Unbiasing involves conditioning the bit-stream, to remove obvious bias that may have been accrued during acquisition and produce an output stream that is more uniformly distributed. There is some confusion on the topic of unbiasing in the IDQ's *Randomness Extraction for the Quantis TRNG* white paper [7]. The latest (2019) version briefly describes Von Neumann processes and alludes to Peres unbiasing methods. However, the precise functioning of the unbiasing algorithm used is not clear. Furthermore, they stress that unbiasing reduces the length of the output sequence, which is not always true, though one must assume that it is for the particular process IDQ implements. It is made clear that unbiasing is a normal part of the output flow of Quantis devices, and that an additional post-processing phase is possible after output is received by a host machine. The output is then forwarded to the interfacing module, which presents the final sequence to the host machine.

In this paper, we refer to Quantis sequences as *raw* or *post-processed*. Raw sequences are those that have been acquired directly from a Quantis device, which unbiases its output by default. An optional (as stated on page 42, point 5 of the user manual [8] post-processing algorithm may be used to sanitise the resulting bit-stream. This has the effect of shortening output files by 25%, as stated in the *Randomness Extraction for the Quantis TRNG* technical documents provided by IDQ. The user manual refers to post-processing as optional, which is known to be incorrect.

The specific processes involved are outlined in the previously mentioned white paper. In brief, a non-deterministic randomness extractor (post-processing algorithm) using a Universal-2 hash function is implemented in the Quantis software package. IDQ provides a sample matrix, though users with sufficient technical expertise can define their own. We use the default post-processing matrix throughout this work.

As the authors of this work, we would like to point out that IDQ's resources on this subject are informative but somewhat disordered. The technical papers for randomness extraction (both the software architecture and mathematical theory) provide a great deal of information, but the product white papers only refer to the onboard unbiasing shown in Fig. 1.

A total failure test is implemented to ensure that hardware failure is detected and output stopped. This prevents a fatally flawed device from outputting highly biased values without the user's knowledge.

IDQ manufactures 4 Quantis generators, 1 of which is AIS-31 [12] certified, and the other 3 we have analysed. They come as PCI-E and USB devices. The 16M and 4M models, shown in Figures 2(a) and (b) possess an output speed of 16 Mb/s and 4 Mb/s, respectively. PCI-E models are available for the 16M and 4M. USB models are only available for the 4M.

Fig. 2. Quantis devices assessed in this research - from left to right: 16M PCI-E, 4M PCI-E, 4M USB

[7] https://www.idquantique.com/resource-library/random-number-generation/#
[8] http://xepa15.fisica.ufmg.br/inetsec/uploadFiles/DOC/User_Manual.pdf

Quantis devices, with the exception of their BSI AIS-31 certified products, have been *self-certified*[9]. The Swiss Federal Office of Metrology (METAS) has also certified the Quantis QRNG range. The Compliance Testing Laboratory (CTL) certified these devices[10]. CTL's certification relates to gambling and gaming applications. IDQ's non-AIS-31 certified QRNG range is marketed towards use in cryptography, gaming, lotteries and similar applications.

In 2014, Krister Sune Jakobsson [13] of Linkoping University identified several issues with the Quantis USB module in their Master's thesis. These results suggest systemic issues with raw Quantis data for the 4M USB model in particular, which has been independently verified by the authors of this paper and extended to a wider range of QRNGs and tests. Of particular note is the poor performance of the Quantis USB module on the TestU01 Alphabits battery [14], which is intended to identify issues in hardware-implemented RNG. This is concerning, as Quantis QRNGs have been used by the Loterie Romande for years. This Swiss lottery generated a revenue of $100 million in 2010 and its business model relies on efficient, random draws of numbers that can be trusted to be unpredictable.

2.2 Comscire's PQ32MU

The PQ32MU is a USB device, which can be accessed through a Linux OS via command-line or an application on Windows. It has an output speed of 32 Mb/s. It does not come packaged with certificates, like the Quantis modules, but the product page does refer to the device being *guaranteed to pass any test for randomness*[11]. It is also stated that the device is designed to meet NIST SP800-90B recommendations. The manufacturers suggest that it is suitable for gaming, security, cryptography and research. This device uses quantum shot-noise as its entropy source.

Wilber [15] stated in 2013 that previously there was not an adequate understanding of how to generate random numbers with a precisely specified or known amount of quantum entropy. The PQ32MU (and other Comscire products) are claimed to have overcome this limitation.

Quantum shot-noise [16] is a phenomenon observed in Metal-Oxide Semi-conductor (MOS) and Complementary-MOS (CMOS) devices, where leakage during use of an electronic device provides a source of entropy from observable suppressed shot noise. The PQ32MU uses this as a source of entropy. The PQ32MU runs regular hardware run-time tests, to determine whether it is functioning within expected parameters. It is advertised as providing 0.99 bits of quantum entropy per bit and is shielded in an aluminium enclosure to protect it against environmental noise that may influence the outcome of the randomness extraction process.

Unlike Quantis devices, this QRNG does not require additional post-processing, claiming to perform all required functions on-device. Comscire also claim that due to this, further post-processing is not required to achieve NIST full entropy.

2.3 Online Generators: Randomness as a Service

The previous devices are physical items, which are purchased and then installed in computer hardware appropriate to the target application. As sources of quantum entropy often require controlled conditions that are difficult to maintain, some academic institutions and individuals have offered randomness as a service, usually free of charge (though often with terms and conditions, such as throttled access speeds).

The Humboldt Physik QRNG is an optical QRNG. It is a joint effort between PicoQuant GmbH and the Nano-optics groups of the Department of Physics, Humboldt University, Berlin. This generator

[9] http://bit.ly/2DdA3rP

[10] Certificate issued 30/03/2011 after successfully passing required tests

[11] PQ32MU product specifications, and certifications can be found at http://bit.ly/2HocFvs

is a high-speed service offering downloads and live numbers to users. A simple command-line application can be used to interact with the generator service and stream numbers into files for testing (hereafter referred to as samples). Wahl et al. [17] claim a speed of 150 Mb/s, with data being post-processed at source then being forwarded to the user.

The Australian National University (ANU), through the efforts of Symul et al. [18], also offers a QRNG service. Broadband measurements of a vacuum field contained within the radio frequency side-bands of a single-mode laser are the entropy source for this QRNG. The observed photo-currents are transformed into a string of random numbers. Symul et al. [18] claim that the possible speed of such a generator is 2 Gb/s, but users experience a far slower rate due to bandwidth and throttling through the service front-end. They also claim to have used a standard battery of randomness tests to verify the output: ANU has tested their generator implementation using Dieharder [19], NIST SP800-22 tests [20], Birthday Collisions [21], and the Dice Probabilities tests.

The official web page for ANU's generator claims that true randomness is generated at a rate of 5.7 Gb/s. This is a result of implementing suggestions made by Haw et al. [22] in 2015 regarding the maximisation of randomness extraction in their implementation. They define true randomness as a sequence in which there is only a 0.5 probability of guessing the next bit in a sequence, which should be completely independent of all preceding bits. They also state, correctly, that their numbers do not have a finite period or seed that can be derived. This is true for all of the generators tested in this work.

We attempted to collect data from the Fourmilab HotBits (RandomX) service[12]. However, issues with the user credentials and extremely low rate of number generation prevented collection of an appropriate sample. Their own statistical analysis shows that this generator passes Dieharder and NIST SP800-22, but it fails a χ^2 test under Fourmilab's own Ent [23] utility[13].

An important concept to bear in mind when considering these generators is the non-deterministic nature of the output. Solitary tests of individual sequences are not sufficient to identify issues; any non-deterministic bit generator (TRNG) will fail a given statistical test at some point due to the inability to guarantee that a given sequence will contain a uniform distribution of values (be they bytes, integers or sub-sequences of arbitrary length). This is a key difference between uniform deterministic (PRNG) and non-deterministic bit generators (TRNG): the former ensure a uniform distribution, while the latter rely on their inherent entropy to provide *truly* random values. A uniform distribution is likely over a given set of sequences, but an individual sequence may exhibit deviations which will cause some statistical tests to fail. Rigorous testing of statistically significant data sets is required to reliably identify recurrent issues from probabilistic phenomena.

2.4 High-speed QRNG

We focus on relatively low-cost, consumer-grade QRNG is this work. ANU and Humboldt are exceptions to this because they are randomness-as-a-service distributors based on devices too large to be practical as a consumer item. Quantis and Comscire generators, however, are representative of consumer-grade QRNG. There are significantly faster, more expensive, and sophisticated devices that have been developed in recent years. Gigabit-per-second devices are not uncommon, though they are not common as commercial items. Such generators are usually built to order, centrally hosted as services which distribute randomness online, or are experimental platforms (with all the size, power and cost issues inherent to prototypes).

[12]http://bit.ly/2HecFhQ
[13]http://bit.ly/2VVKlUE

Zheng et al. [24] propose a QRNG based on vacuum-fluctuation observations, which can reach output speed of 6 Gb/s of reliably random values. This device relies on Field-Programmable-Gate-Array (FPGA) implemented Toeplitz hashing [25] to achieve high-speed, real-time post-processing. Zheng et al. state that their generator has a theoretical maximum speed of 12 Gb/s, but that information-theoretic randomness is only achievable at speed of 6 Gb/s. Xu et al. [26] achieve similar speeds also using optical methods.

Due to advancements in FPGA-assisted self-testing and post-processing techniques, these generators are rapidly becoming viable for commercial production. However, as the cost of early adoption is significant, it is important to identify the security benefits of these new devices, over existing devices. In this work, we analyse current highly-adopted devices (in the 4-32 Mb/s range) in the interests of evaluating current certification and security testing schemes.

3 METHODOLOGY

3.1 Data collection

All QRNG-derived data was collected and tested on an Ubuntu workstation with an Intel i7 3770k processor and 16GB of RAM.

The Quantis *Easy Quantis* Linux command-line application was used to collect 2GB from each Quantis device. The processes outlined in the Quantis user manual were followed, collecting both raw data and post-processed data.

A total of 100 samples were collected for each QRNG. As a result of compression during post-processing, separate files of 1.7GB in size were created, a reduction of 25%. This is in line with the values specified in IDQ's *Randomness Extraction for the Quantis TRNG* white paper.

Post-processed data is tested alongside the raw data, to demonstrate the importance of post-processing to QRNG operation and highlight flaws in IDQs advice to consumers. It should be noted that post-processing is a time-intensive process, taking approximately 12 minutes for a 2GB file on the previously specified PC. A benchmarking experiment showed that post-processing over the live stream of bits from a Quantis device cut the speed of data acquisition by up to 75%, with a minimum speed-reduction of 58%. This is above the 25% stated in the IDQ *Randomness Extraction* white paper. The size of files post-processed in this manner was reduced by the 25% figure stated in the aforementioned document.

The Comscire PQ32MU's output stream was read for 2GB blocks into 100 files using the *dd* utility. As the device was read in its default mode, the output was post-processed on-device, prior to our data collection efforts. A raw stream was not collected.

The data collection method for both the ChaosKey and *urandom* involved the use of the *dd* utility to print random values from each device to a binary file. Sequences of 128 bytes were extracted 16,406,250 times to achieve a final file size of 2GB for both RNGs. This is a significant difference when compared with the Quantis devices, which provide an application for data extraction, while the ChaosKey and *urandom* are both supported by the Linux kernel. The Quantis devices also require driver installation and a roll-back to kernel v3 otherwise they will throw a 'version not recognised' error when compiling. The operating system of the host machine was Ubuntu 16.04 LTS. The Quantis devices, *urandom* and the ChaosKey were tested with the v3 of the kernel, which is required by the version of Quantis software used.

The Humboldt Physik generator has sufficient speed to serve *live* random numbers quickly enough to allow timely collection of a 2GB sample. The ANU generator is significantly slower, and therefore our sample size is based on static data drawn from the generator and supplied by ANU researchers. The small size (approximately 100MB) of these samples prevents the use of some tests over the data.

3.2 Randomness tests

The outputs of the Quantis devices, Comscire PQ32MU, ANU QRNG, and Humbolt Physik QRNG were subject to 5 batteries of statistical tests. Where possible, QRNGs were tested using Dieharder, NIST SP800-22, Ent, TestU01 (Alphabits, Rabbit, Small Crush and Crush), and Tuftests. TestU01 Small Crush and Crush were not run over the ANU QRNG, as insufficient data was collected to satisfy the requirements of said test batteries. The Randomness-Testing-Toolkit (RTT) statistical testing tool[14] was used to run selected TestU01 batteries, and Dieharder. The Ent utility packaged with Ubuntu 16.04 LTS was used for Ent tests. NIST STS-Master[15] (an updated version of the NIST SP800-22 tests) was used for the NIST tests, allowing for faster execution of these tests.

Each statistical test shares a null hypothesis (H_o): *The tested data does not exhibit signs of order or structure.* This common H_o means that all tests share the following definitions for type 1 and 2 errors:

- Type 1: The sample fails the test erroneously. This will happen (eventually) with some small probability depending on the test and its parameters.
- Type 2: The sample passes the test erroneously.

3.2.1 Dieharder. The Dieharder battery extends the original Diehard tests with the NIST SP800-22 tests. It is considered a good gauge of the general health of a generator, with failure on any test being an indication of a flawed RNG. Passing this battery doesn't guarantee randomness, but consistent failure is an almost-sure sign of non-randomness. It is a part of many certification test strategies, including BSI's AIS-31.

The Quantis devices, Comscire PQ32MU, ANU, and Humbolt generators were analysed using Dieharder. All tested samples were tested using Dieharder for their full length, without rewinds. Due to ANU's smaller sample size, only 23 tests met these requirements. For the Quantis, PQ32MU and Humboldt Physik QRNGs, the *1000MB.json* RTT configuration file was used for Dieharder. ANU's QRNG was tested using the *100MB.json* configuration file. These files are available on the RTT Github page (see footnotes). Non-default test parameters are outlined in detail in the Appendix.

3.2.2 NIST SP800-22. NIST offers the Statistical Testing Suite (STS) defined in SP800-22 [20] for the analysis of randomness. This software provides a more intuitive user interface than Dieharder and allows the size and number of bit-streams to be defined. It is possible to select which tests are used, but for this research, all tests have been employed.

All samples except the ANU QRNG have been analysed for a stream size of 1,048,576-bits and 2000 bit-streams. ANU's QRNG has been tested for 128 bit-streams (of size 1,048,576-bits) to accommodate the smaller file size. NIST documentation suggests at least 100 sequences are tested, stating that a higher number of sequences increases the accuracy of the test. This is reinforced by Marton et al. [27], who refine the definition of a passing sequence to include a more accurate value for the number of failed uniformity and proportion tests that can be tolerated. Applying the NIST advised confidence level of $a = 0.01\%$ allows use to tolerate up to 2 uniformity and 4 proportion test failures.

3.2.3 Ent. Walker's [23] Ent utility evaluates pseudo-random number generators using 5 tests; Entropy, compression, χ^2, arithmetic mean, Monte Carlo value for π and serial correlation coefficient. The most important tests for this work are the χ^2 test [28] and serial correlation coefficient. The former determines whether the occurrence of values (of n-bits) are uniformly represented within a sequence. The latter tests for relationships between an n-bit value and the lagged version of itself,

[14]http://bit.ly/2DcM2Wm
[15]http://bit.ly/2TPOl7i

the presence of a predictable relationship between these two variables indicates non-randomness. In our previous work [29], we have found Ent to be an under-rated bias detection suite. Simple χ^2 tests are able to identify egregious biases that other tests fail to highlight, a fact that casts doubt on test selection criteria for contemporary RNG validation schemes.

Ent is not part of any recommended RNG testing strategy but we [30] have used it to find significant and persistent biases in the DESFire EV1 RNG in previous work. This battery has been applied to all data collected during this research. All samples were analysed for their full length.

3.2.4 TestU01. TestU01, implemented by L'Ecuyer and Simard [14] in the ANSI C language, is a battery of statistical tests intended for use on uniform random number generators. Re-implemented versions of standard tests are included, alongside newer tests implemented from the literature by the developers. Four of the batteries provided in this tool are used in this work: Alphabits, Rabbit, Small Crush, and Crush. The first two are focused on binary sequence testing, while the latter two take 32-bit unsigned integers as their input. All tests were executed in their default parameters, with the exception of Crush. Crush was executed using a reduced set of tests to accommodate our maximum file size of 2GB. Big Crush was excluded, as the RTT battery doesn't support the execution of this test on 2GB files. The Appendix contains the detailed test parameters for Crush.

L'Ecuyer and Simard [14] succinctly state that the number of tests that can be applied to the problem of determining whether a sequence is random is vast, and all but a tiny fraction are too complicated to be implemented. This has led to the common sense statement that good generators only fail overly complicated tests and bad ones only fail tests that are simple enough to implement. This makes sense, as a generator will eventually fail any test if the test in question is sufficiently rigorous. As a result, this work focuses on generators that consistently fail so-called *reasonable* tests of randomness, rather than seeking specific failures that occur only in the most rigorous of tests.

These tests have been applied over the full length of each of the samples taken during data collection. Due to insufficient sample size for ANU QRNG samples, Small Crush and Crush have not been used to analyse that service.

3.2.5 Tuftests. The Tuftests battery consists of 3 statistical tests that Marsaglia and Tsang [21] have deemed *hard to pass*. The tests analyse the statistical properties of (hypothetically) independent and uniform sequences of 32-bit integers. The developers of this battery state that many RNGs will pass other tests of randomness, and that this is sufficient for their use in a general sense. However, if generators are implemented for use in cryptography and security, it is advisable that they are able to pass these tests.

The 3 tests used in this battery are the Greatest Common Divisor (GCD) test [21], Gorilla test [31], and Birthday Spacings test [32]. The GCD test compares paired sequences, using Euclid's algorithm to determine the GCD of the 2 sequences. Monkey tests involve the production of sequences of *letters* selected from an alphabet, before studying the frequency of k-letter words in that sequence. Marsaglia and Tsang [33] devise a more stringent version, the Gorilla test, which involves selecting one of 32-bit positions from integers produced by a generator, and collecting $2^{26} + 25$ of these bits. Each of the 32-bit positions is tested. A p-value is derived from the number of 26-bit *words* (x) that do not appear in each sequence. The Birthday Spacings test involves the selection of m birthdays from a year of n days. After sorting, the number of duplicate values among the spacings of the ordered birthdays should be asymptotically Poisson distributed with parameter $\lambda = m^3/(4n)$ [21]. This version uses more demanding parameters than those used in Diehard (4096 birthdays, each a 32-bit integer representing a day in a year of 2^{32} days).

These tests were applied using default parameters to all samples except the ANU QRNG, which ran the GCD test at 10^6 iterations instead of the 10^7 used for all other samples. Default parameters

test 161 mebibytes (miB) of data, and the ANU QRNG parameters tested 89 MiB, to respect its smaller file size.

3.2.6 Beyond statistical testing. Statistical tests are the focus of this work, but the evaluation of RNGs is not restricted to these tests. Bundesamt für Sicherheit in der Informationstechnik (BSI) provide guidelines for RNG evaluation in AIS-31 [34]. These requirements state that a stochastic model of the entropy source is required for evaluation. IDQ do provide an AIS-31 validated RNG as part of their product-line, with a far lower output speed (7kbps as opposed to 4 Mb/s or greater). For this, we assume they have provided such a model. Unfortunately, due to the black-box design and completely sealed hardware of the QRNGs we have tested, we have not been able to construct a model that matches the specifications of the QRNGs tested here. Reverse engineering of these generators to be able to perform this task is a priority in on going work.

NIST SP800-90B provides both guidelines for the theoretical evaluation of an entropy source, and further tests of minimum entropy [35]. The issue of closed-design in the QRNGs tested remains for the theoretical evaluation of the entropy source, but SP800-90B does provide another entropy-estimation battery. However, we are currently conducting an evaluation of this entropy estimation battery based on the criticisms levelled by Zhu et al. [36], and deemed the discussion of underestimation in the compression and Collision to be divergent from the focus of this work; the evaluation of QRNGs using commonly used statistical test batteries.

4 RESULTS

In this section we present each test battery's results. These are discussed in their respective subsections, with analysis of exceptional results following after. The length of most test reports makes presenting them within page constraints highly problematic. As a result, the tables and graphs below will show the number of tests passed out of the total number of samples for each RNG.

4.1 Benchmarking

The Quantis and Comscire QRNGs selected for this research have had their mean output speed over all collected samples recorded. The ANU and Humbolt QRNGs have been omitted as their output speeds are constrained by bandwidth controls set by their respective operators, rendering any comparison with devices that we have physical control over unfair. Post-processed output from Quantis QRNGs has also been omitted, as this depends solely on the processor speed of the host machine. Table 2 provides the results of the benchmarking phase of this research. For comparison, the ChaosKey TRNG has an output rate equivalent to that of full USB (8 Mb/s).

Table 2. Number, size and speed of samples collected from target RNGs

	No. Samples	Sample Size MB	Mean data rate (Mb/s)
Quantis 16M	100	2100	15.87
Quantis 4M	100	2100	3.86
Quantis USB 4M	100	2100	3.96
Comscire PQ32MU	100	2100	30.99

These results confirm that the devices function close to their marketed speeds, and well within the 10% bound specified in the product specification document. The maximum time taken for the 4M PCI-E and USB models to provide a 2 GB sample was 1 hour and 7 minutes. By comparison, the

16M generated 2GB of data in approximately 17 minutes. Similar values were achieved during our pilot study involving data extraction using the *Easy Quantis* application on Windows 10.

Neither of the online services has been reliably bench-marked, as connection speeds to the service were found to be variable based on network load and route between the authors and the host generator. As a result, these statistics are not reported as they are not reflective of the on-site data-rate of the 2 QRNG services in question.

4.2 Dieharder, NIST, and TESTU01

Table 3. Dieharder, NIST SP800-22 and TestU01 Results

Device	Samples #	Dieharder Passed	NIST Passed	Alphabits Passed	Rabbit Passed	Small Crush Passed	Crush Passed
16M PCI-E	100	100	100	54	60	93	47
Post 16M PCI-E	100	100	100	95	87	91	82
4M PCI-E	100	100	100	3	7	91	3
Post 4M PCI-E	100	100	100	91	82	93	86
4M USB	100	100	100	3	21	89	3
Post 4M USB	100	100	100	90	81	97	80
Comscire PQ32MU	100	100	100	91	86	93	84
ANU QRNG Server	10	10	10	9	8	-	-
Humboldt Physik	10	10	10	10	8	7	10

Table 3 outlines the results generated using three well-known statistical test batteries, intended for use in ascertaining whether data is sufficiently random. Each set of results represents the number of passing samples for each test battery, out of a total number of samples (2nd column from the left).

4.2.1 Dieharder. The Quantis 16M PCI-E module reported at most 3 weak results for a single file across 100 samples. In each case, it was the RGB Lagged Sums test that reported weak results. The 16M didn't fail any tests, despite these weak results. Results for the 4M PCI-E model were much the same: up to 7 weak results but no failures. Unlike the 16M, these weak results were scattered across a number of tests. The only notable recurrence was the STS Monobits test, which reported weak results for 7 of the samples. The 4M USB reported a maximum of 9 weak results but passes all tests across 100 samples. This indicates that these generators have passed the Dieharder test suite. There were no statistically significant findings of non-randomness. Post-processed versions of each file passed the Dieharder battery. No tests reported weak or failed results for post-processed data.

Similarly, the Comscire PQ32MU passed all Dieharder tests. Fewer weak results (at most 2 and only for 4 samples) were observed throughout the 100 samples tested than were reported by the Quantis 4M (both USB and PCI-E models). ANU QRNG and Humboldt Physik both passed this test battery with no weak or failed tests. ANU's small file size prevented a full analysis using the whole suite of tests. This emphasises the importance of testing on a sufficiently large sample of data.

4.2.2 NIST SP800-22. The Quantis 4M PCI-E model and 16M reported some borderline results for Non-Overlapping Template and Random Excursions tests. As discussed previously, more than 2 failed tests of uniformity, and 4 failed tests of proportion are required to determine whether a generator exhibits non-random behaviour. None of the test results reported uniformity or proportion failures sufficient to fail a test.

In most Quantis 4M USB samples a Non-overlapping test of proportion was failed, but not to any great degree. As with the previous Quantis PCI-E devices, the test failures varied between samples, without consistent repetition of test statistics. As there are insufficient uniformity or proportion test failures to exceed Marton et al.'s [27] threshold, this device passed the NIST battery. Post-processing

has a negligible effect on the outcome of this test, reducing the number of failures of proportion and uniformity for each test. This is expected, as a sequence that passes in its raw form is unlikely to fail after post-processing. A sequence that fails these tests may pass after post-processing, however.

Comscire's PQ32MU passes all SP800-22 tests. The maximum number of failed uniformity tests is 1, and proportion tests fail at most 2 times. The total number of failed tests throughout the whole sample is comparable to that of the post-processed 16M data. This generator passes the NIST SP800-22 battery. ANU and Physik similarly pass the battery.

4.2.3 TestU01 Alphabits. Tables 4 and 5 provide detailed results for the Alphabits battery of tests. In this and all subsequent TestU01 tables, **bold** text indicates a result of particular interest that is discussed in the text.

Table 4. TestU01 Alphabits failure rate for Quantis devices

	16M	Post16M	4M PCI-E	Post 4M PCI-E	4M USB	Post 4M USB
Total samples	100	100	100	100	100	100
smultin MultinomialBitsOver	**46**	2	**97**	8	**92**	8
sstring HammingIndep	2	0	2	1	2	0
sstring HammingCorr	1	3	0	2	0	2
swalk RandomWalk1	5	1	9	0	6	0

Table 5. TestU01 Alphabits failure rate for ANU, Humboldt and PQ32MU

	ANU	Humboldt	PQ32MU
Total samples 100	100	10	10
smultin MultinomialBitsOver	1	0	4
sstring HammingIndep	0	0	2
sstring HammingCorr	0	0	2
swalk RandomWalk1	0	0	1

TestU01 is where the Quantis generators began to show significant issues. The 16M, and 4M (PCI-E and USB) fail the Alphabits battery. Up to 3 of 4 tests (and 11 of 16 statistics) failed. All 3 Quantis devices failed the Multinomial Bits Over test repeatedly, with 46 failures for the 16M, 97 for the 4M PCI-E and 92 for the 4M USB. Comparing these results with those in Table 3 shows that almost every test failed by each raw stream from the Quantis devices has a failed Multinomial Bits Over test associated with it. There are some failures in other tests, but these account for less than 10% of the samples for each device. It is clear to see that the Quantis devices failed this battery for their raw data.

This degree of failure in the MultinomialBitsOver indicates an imbalance in the distribution of 1's and 0's within observed sequences [37]. In some TRNGs, start-up biases can contribute these characteristics to an otherwise random sequence. This was ruled out by ensuring that all sequences generated for these experiments were collected after an initial period of QRNG output, specifically a 10 MB pre-test sequence that was removed from the sample prior to testing. Cursory testing of the original sequences, including the removed 10 MB, provided the same failure rates as shown in Table 4. This demonstrates that not only was correct procedure to avoid such bias upheld, but even if it had not, start-up biases are not the contributing factor here. Further analysis of the test

sequences is required to identify where, specifically, the bias lies. There was no discernable pattern to the bias within individual sequences. Some would demonstrate a predominance of overlong runs of 1, others 0. The distribution of biases to 1 or 0 revealed no statistically significant deviation towards one of the other in these experiments. A larger dataset and more rigorous examination of sub-sequences within the samples will be required to identify any such characteristics, should they exist.

Post-processed Quantis data performed better but still failed up to 10% of the time. In the 4M PCI-E and USB, 8% of tests failed due to the Multinomial Bits Over test, but post-processed 16M data shows a different pattern, with no instances of multiple tests having failed and a majority of 3% of the sample failing the Hamming Correlation test. It is clear that post-processing improved the situation, but there were still failures over 5% of the tested samples.

The PQ32MU was comparable to post-processed Quantis data, with a failure rate between that of the 4M PCI-E, 4M USB, and 16M devices. Multinomial Bits Over was the most failed test with 4% of samples failing it. Samples reported multiple failures, only one test was failed in each case.

Humboldt's Physik generator passed the battery. ANU QRNG fails 1 Multinomial Bits Over test. Further testing on a more substantial number of larger sample samples will be needed to obtain a more significant result.

Table 6. TestU01 Rabbit failure rate for Quantis devices

	16M	Post 16M	4M PCI-E	Post 4M PCI-E	4M USB	Post 4M USB
Total samples	100	100	100	100	100	100
smultin MultinomialBitsOver	11	6	6	11	6	12
snpair ClosePairsBitMatch	7	0	3	4	4	4
svaria AppearanceSpacings	0	2	0	2	0	1
scomp LinearComp	0	1	0	0	0	1
scomp LempelZiv	2	0	1	2	1	1
sspectral Fourier1	0	0	1	0	0	0
sspectral Fourier3	0	0	**42**	0	**36**	0
sstring LongestHeadRun	2	0	1	0	0	0
sstring PeriodsinString	0	2	0	0	1	0
sstring HammingWeight	15	1	14	0	8	3
sstring Hamming Corr	3	1	0	0	0	0
sstring HammingIndep	3	0	2	0	1	0
sstring AutoCor	2	1	**86**	0	**81**	1
sstring Run	3	0	**67**	0	**61**	1
smarsa MatrixRank	0	1	0	2	0	0
swalk RandomWalk1	3	1	3	1	1	0

4.2.4 TestU01 Rabbit. Tables 6 and 7 show the detailed results for the Rabbit battery.

Raw data from all 3 Quantis devices performed very poorly in this battery, but the characteristics of the failures were changed compared to Alphabits. Multinomial Bits Over is run with different parameters: $L = 38$ instead of 2, and $n = 9998336$ instead of 932067552. The Multinomial Bits Over test is an overlapping test, sequences are selected of length L, for a total of n sequences, shifting by 1-bit for each sequence selection. The size of output collected from the target generator (or file) remains $s = 32$ and L does not need to be divisible by s for the overlapping version of this test. The Rabbit implementation of this test reported fewer failures. This is almost certainly because the tested sequences were longer, resulting in fewer collisions between sequences being detected.

The 4M PCI-E and USB exhibited a concentration of failures in the Run (up to 86%) and Auto Correlation (up to 67%) tests. The Fourier3 test also reported 42% and 36% failure rates for the 4M and USB respectively. The 16M didn't share this association, instead reporting a greater number

Table 7. TestU01 Rabbit failure rate for ANU, Humboldt and PQ32MU

	ANU	Humboldt	PQ32MU
Total samples 100	100	10	10
smultin MultinomialBitsOver	0	2	10
snpair ClosePairsBitMatch	0	0	2
svaria AppearanceSpacings	0	0	0
scomp LinearComp	0	0	2
scomp LempelZiv	0	0	1
sspectral Fourier1	0	0	0
sspectral Fourier3	0	0	0
sstring LongestHeadRun	0	0	1
sstring PeriodsinString	0	0	0
sstring HammingWeight	0	0	0
sstring Hamming Corr	0	0	0
sstring HammingIndep	1	0	1
sstring AutoCor	0	0	0
sstring Run	0	0	1
smarsa MatrixRank	1	0	0
swalk RandomWalk1	0	0	0

of Multinomial Bits Over (11%) and Hamming Weight (15%) failures. Multiple test failures were common for all devices, but dramatically more so for the 4M PCI-E and USB, which reported at least 2 different test failures for over half of their failed samples.

Post-processing once again reduced the number of failures dramatically, and removed the trend of Auto Correlation and Run failures under the 4M PCI-E and USB. This demonstrates that post-processing significantly alters the data in question. However, it still couldn't pass the battery and fails for at least 12% of samples. Strangely, the trend of failing the Multinomial Bits Over test returned after post-processing. Multiple test failures were dramatically decreased, with fewer than five samples showing more than 1 test failure.

The PQ32MU maintained parity with the post-processed output of Quantis devices. Like the post-processed data, it had the most trouble with the Multinomial Bits Over test, with 10% of samples reporting failures in this test. Multiple test failures for individual samples were uncommon ($< 3\%$).

Both Physik and ANU reported 2 battery failures with a single test failing in each. This is proportionally equivalent to the performance of post-processed Quantis and PQ32MU data. More data will be required to make a more accurate comparison, but it appears that all post-processed QRNG output exhibits better performance on this battery. As easy at it would be to write off these failures as type 1 errors, they are still too high for comfort. Some individual test failure rates possessing a greater than 1% probability.

4.2.5 TestU01 Small Crush. Table 8 shows the full results for the Small Crush battery.

Before discussing these results, it is important to understand how the Crush tests process their input data. All three of the Crush tests take floating point numbers as input. As with the other tests, sequences of 32-bits are drawn from an input source. In the case of the Crush tests, these 32-bit sequences are converted into a list of floating point numbers prior to test initialisation. In this work, this was achieved using a simple Python script.

All Quantis generators performed well on this battery, compared to previous results. However, most still reported a failure rate of over 5%. Interestingly, the post-processed 16M samples failed more frequently than the raw 16M data. There's no indication of why this occurred, though it is possible that the redistribution and compression of values may randomly result in a post-processed

Table 8. TestU01 Small Crush failure rate

	16M	Post 16M	4M PCI-E	Post 4M PCI-E	4M USB	Post 4M USB	Humboldt	PQ32MU
Total samples	100	100	100	100	100	100	10	100
smarsa Birthday Spacings	2	1	0	0	1	0	0	4
sknuth Collisions	0	2	2	2	1	1	0	0
sknuth Gap	1	1	1	0	1	0	0	0
sknuth SimpPoker	1	0	1	0	1	2	2	0
sknuth CouponCollector	1	1	0	0	2	0	0	0
sknuth Max0ft	1	3	0	1	1	0	0	0
svaria WeightDistrib	1	0	3	1	4	1	0	0
smarsa MatrixRank	1	0	0	1	0	1	0	0
sstring Hamming Indep	0	0	1	1	1	2	0	3
swal RandomWalk1	0	0	1	2	0	1	1	0

file that fails more tests than the original data. The 4M PCI-E and USB did not exhibit this behaviour, reporting fewer test failures for post-processed data. The PQ32MU had a failure rate similar to the Quantis devices (and their post-processed output) for this battery. None of the physical QRNGs displayed significant bias towards one test as a source of failures.

Physik failed for 3 out of 10 samples. Two of these failures were observed in the Simple Poker test, with 1 in the Random Walk 1 test. There are no cases in which multiple tests fail for a single file.

4.2.6 TestU01 Crush. The full results can be found in Table 9.

Table 9. TestU01 Crush failure rate

	16M	Post 16M	4M PCI-E	Post 4M PCI-E	4M USB	Post 4M USB	Humboldt	PQ32MU
Total samples	100	100	100	100	100	100	10	100
smarsa CollisionOver	3	3	3	3	5	4	0	5
smarsa BirthdaySpacings	0	1	1	2	0	1	0	1
snpair ClosePairs	0	1	2	0	0	0	0	3
snpair ClosePairsBitMatch	7	5	8	8	5	6	0	2
sknuth Max0ft	0	1	0	1	2	0	0	3
svaria SampleProd	0	2	0	2	1	1	0	0
svaria SampleMean	11	0	9	0	8	1	0	1
svaria AppearanceSpacings	2	1	0	0	1	0	0	0
smarsa MatrixRank	2	1	2	0	0	1	0	0
smarsa GCD	1	0	1	0	2	1	0	2
swalk RandomWalk1	14	2	7	1	8	2	0	1
scomp LinearComp	1	2	0	0	2	0	0	0
scomp LempelZiv	1	2	1	0	0	1	0	1
sspectral Fourier3	0	0	13	1	5	1	0	0
sstring HammingIndep	2	1	0	0	1	1	0	0
sstring Run	**38**	0	**97**	0	**97**	1	0	2

The Crush battery is a more robust 32-bit sequence testing battery than Small Crush, implementing 16 tests with multiple repetitions and sub-tests. Raw Quantis data performed exceptionally poorly on this battery, much as it did for Rabbit. The 4M PCI-E and USB passed for only 3 of 100 samples each. The 16M passed 47% of the time for raw data, making this the worst performing battery for raw Quantis data. For the 4M PCI-E and USB models, the trend of failing Run tests is carried over from Rabbit. Interestingly, this trend was observed in the results of raw 16M data as well. 38 out of 53 battery failures also reported a failed Run test.

Post-processing reduced the number of failures, but over 14% of Quantis samples still fail the battery even after post-processing. The concentration of test failures within failing samples is dramatically reduced. At most 8 out of 14 failed samples reported the same test as an issue (in this case the Random Walk 1 test for post-processed 4M data).

As with all previous examples, the PQ32MU demonstrated similar properties to the post-processed Quantis data. It failed 16 of 100 samples but exhibited a more even distribution of failures throughout the possible configurations of failed tests. Rarely were more than 2 tests failed in a single sample.

Humboldt Physik passes the Crush battery with no issues.

4.3 Tuftests

Table 10. Tuftests Results

Device	Sample #	Birthday Spacings Passed	Steps to GCD Passed	Dist. of GCDs Passed	Gorilla Passed
16M	100	100	65	100	100
Post 16M	100	100	66	100	100
4M PCI-E	100	100	56	100	100
Post 4M PCI-E	100	100	70	100	100
4M USB	100	100	68	100	100
Post 4M USB	100	100	77	100	100
Comscire PQ32MU	100	100	99	100	100
ANU QRNG	10	10	10	10	10
Humboldt Physik	10	10	6	10	10

Tuftests results are shown in Table 10. All devices passed the Birthday Spacings, Distance of GCD and Gorilla tests. This is at odds with some of the previous results: NIST SP800-22 and TestU01 implement Birthday Spacings and GCD tests as a part of their batteries. However, the configurations of those tests are dramatically different. For NIST SP800-22 and TestU01, the full length of each sample file has been used for the tests in question. For Marsaglia's Tuftests, the default 161MB of data has been used (89MB for ANU samples). This explains the difference in results, and why many tests that see occasional failures in tests using larger file sizes are not replicated here.

A significant proportion of samples for Quantis devices and Humboldt's Physik QRNG failed the Steps to GCD test, with a confidence value of 0.01. Post-processing was found to only marginally improve this for Quantis results. Every generator that failed this test did so with a value of $p > 0.99$.

The PQ32MU reports a worrying number of failures in TestU01, but passed Dieharder, NIST and Tuftests. The Quantis devices were shown to be weak in both TestU01 and Tuftests, contradicting the positive impression given by Dieharder and NIST SP800-22. This carried over to post-processed data, which appeared to have eliminated some issues with raw Quantis data, but exhibited weaknesses in many of the same tests. This may indicate that the tests are overly sensitive, reporting a large number of false positive for non-randomness, or that the algorithm used to post-process the Quantis data is not as robust as that used by Comsire's PQ32MU. Considering the results shown in Table 10, the latter is quite likely, at least for some tests.

4.4 Ent Results

The Ent battery provides χ^2, serial correlation coefficient, arithmetic mean, Monte-Carlo π, and entropy tests. Of these, the χ^2, and serial correlation coefficient are reported in this section, for byte and bit-level analyses. A confidence level of 0.05 was selected for these tests. p-values for the serial correlation coefficient results were calculated by computing Student's t-distribution for each

result, then determining whether each result exceeded the critical-value, as outlined by NIST[16]. All devices pass the other tests for all of their samples.

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \tag{1}$$

The t-distribution (Eq. 1) of the serial correlation coefficient (r) is computed to better identify whether the resulting value is in or out of expected bounds for a given input sequence. Where the t-distribution exceeds 1.96 (p-value = 0.975), one should reject the null hypothesis. Such a result indicates a correlation between the bytes (or bits) of the sample. The degrees of freedom (DOF) used in the calculation of the final p-values from the t-distribution were 255 for byte level tests, and 1 for bit level tests.

Table 11 displays the number of samples that passed the χ^2 and serial correlation coefficient tests for each device. The χ^2 test used in this battery takes individual bytes, in sequence and maintains a count of all bytes of a given value. The score derived from this process can be used to calculate the p-value of the tested sequence, where $p \leq 0.01$ is considered a failure for the purposes of this work. The Quantis devices are separated into raw and post-processed data, while all other sources provide a single entry.

Table 11. Ent results

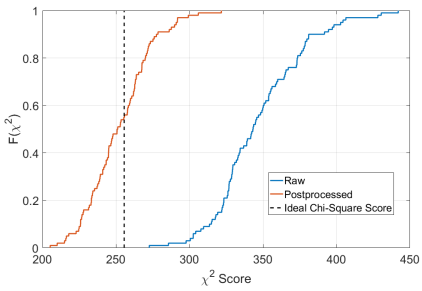
		Bytes		Bits	
Device	Samples #	χ^2 Passed	Serial Corr. Passed	χ^2 Passed	Serial Corr. Passed
16M	100	10	99	0	100
Post 16M	100	100	96	100	96
4M PCI-E	100	0	99	0	49
Post 4M PCI-E	100	100	99	100	100
4M USB	100	0	92	0	81
Post 4M USB	100	100	94	100	100
Comscire PQ32MU	100	100	99	100	100
ANU QRNG	10	10	8	10	10
Humboldt Physik	10	10	10	10	10

Quantis devices struggled with the serial correlation test, especially at the bit-level. At the byte level, not much difference was observed between raw and post-processed results, with > 92% of samples passing the test. However, at the bit-level, raw data from the 4M PCI-E failed 51% of the time. The 16M failed more when post-processed than for raw data under this test, the reasons for which are unclear but could be related to structural changes in the resulting data. Post-processing removed all issues for the 4M PCI-E and USB devices at the bit-level, resulting in a 100% success rate.

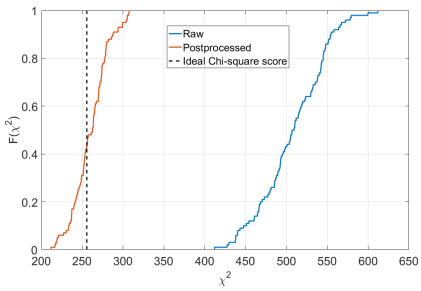
Raw data from the Quantis devices, barring 10 exceptional 16M results, failed the χ^2 test. They reported excellent results for every other test, passing each. However, the χ^2 scores and p-values are exceptionally high. Tests on smaller sample sizes found that many of these devices passed the χ^2 test at sizes of under 500MB (some even at 1GB or less).

Figure 3 shows the cumulative distribution function (cdf) of byte-level χ^2 test scores for raw and post-processed Quantis data. As can be seen in all three graphs, the post-processed data centres on the ideal χ^2 score, while raw data far exceeds that value.

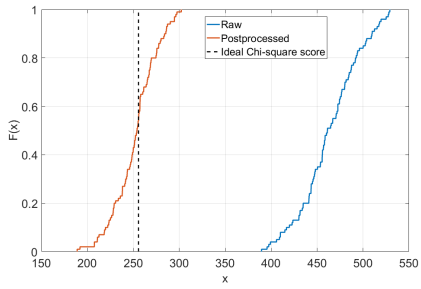
[16]http://bit.ly/2Dcuo56



(a) 16M



(b) 4M



(c) USB

Fig. 3. Cumulative distribution function of χ^2-scores for Raw and Post-processed data

The 16M (Fig. 3a) performed best out of the three Quantis devices, as can be seen by the smaller distance between the raw and post-processed cdf results. It failed all but 10 χ^2 tests. The passed test is borderline, with a score of 305.13 and a p-value of 0.07. This is barely a pass and is a cause for concern. The average score for its raw data was 347.33 and for post-processed data was 252.40.

The 4M PCI-E (Fig. 3b) and USB (Fig. 3c) models had the worst performance. Both reported scores over 500, with their lowest scores being 450.17 (4M PCI-E), and 404.02 (4M USB). The 4M reported an average score of 507.76 for raw data and 260.41 for post-processed data. The 4M USB model reported an average score of 463.07 for raw data and 252.34 for post-processed data. This demonstrates the ability of post-processing to improve the uniformity of data, though the failure of post-processed data in some of the other batteries used in this work suggests that weaknesses may remain (or may be introduced).

Previous research undertaken by the authors [29], has identified that seemingly robust RNGs can fail on these most straightforward and often overlooked tests. This is especially true of hardware random number generators. A poor χ^2 result indicates that there is a low degree of uniformity among byte values in the sample. The occurrence of values may be clustered around a specific point, or a pattern may have emerged, leading to a small but detectable structure in the frequency with which given byte values occur.

In our previous work with the DESFire EV1 TRNG, this took the form of a sinusoidal structure, in which half of the possible byte values occurred fractionally more frequently than the other half, leading to a perceptible trend in the plotted occurrences of said bytes.

5 DISCUSSION

We have found that although Quantis devices in raw output mode tend to pass Dieharder and NIST SP800-22 tests, they fail the χ^2 test of the Ent suite significantly. The testing regimen that the sampled devices have been subjected to has been expanded to include TestU01 and Tuftests. Samples that appear robust under Dieharder or NIST SP800-22 fail catastrophically under several TestU01 batteries. TestU01 has demonstrated the effectiveness of post-processing to sanitise the low-quality raw output of Quantis devices, while Tuftests and TestU01 (Rabbit and Crush batteries) show that there are still issues in the resulting output that must be addressed.

METAS [38] only really certifies that the Quantis modules pass the Diehard tests for 10 samples of 100MB. Quantis verifies these devices in-house[17], providing a certificate that states they passed the Compliance Testing Laboratory (CTL) [39] tests. To their credit, IDQ tested over 1 billion bits, approximately 125MB. This conforms with the minimum expectations of NIST SP800-22. However, this is still well within the range of sample sizes that we have observed as passing tests even when larger samples will systemically fail.

The difference between the biases in each Quantis module sample suggests that the issue is not reproducible in any single form, but instead, a unique set of biases is generated in each sample. This shows that there is a degree of randomness provided by these devices, likely sufficient to appear random when tested under traditionally accepted suites such as NIST SP800-22, and Dieharder.

The byte-level analysis shows that there is a consistent degree of failure and associated levels of bias, despite the variable manner that said bias presents itself between samples. Further analysis will be required to characterise the bias shown by Quantis generators and identify whether the thermal-noise interference posited by IDQ in their data-sheet for each device is accurate. If thermal-noise dominates the raw output of these generators, that would explain some of the findings presented here.

The Comscire PQ32MU has some poor results for the TestU01 battery, but exhibits no trend for failure in specific tests, making any further identification of issues with the generator difficult. Worryingly, Comscire state that they performed Diehard and NIST SP800-22 over 80Mb (80,000,000 bit) samples[18]. This is insufficient according to the guidelines of both tests, NIST stating that at least 100 bit-streams of over 1,000,000 bits are required for completion of the battery. This work suggests that NIST recommendations may be too lenient and that larger and more numerous samples must be collected. The ANU and Physik generators require larger samples for a more representative analysis, but initial results indicate that they perform as well as PQ32MU and post-processed Quantis data.

An important item to note when considering Quantis and Comscire generators is that the Comscire generator implements its post-processing algorithm in the device itself (as do Physik And ANU). Quantis generators forward raw output by default, requiring software post-processing off-device. This is why we have identified Quantis generators as particularly weak; their post-processing implementation is itself flawed and deemed *optional* in IDQ's own documentation [19], point 5 on page 52. This is extremely bad advice, given the results presented in this paper. By implementing post-processing in software, the attack surface of Quantis devices is increased significantly. A malicious actor may even be capable of using the post-processing software to implement a software-Trojan that injects random-seeming but exploitable sequences into the resulting output. This possibility is an item of work we are currently exploring.

[17] http://bit.ly/2RrmE89
[18] http://bit.ly/2CsLpX0
[19] http://xepa15.fisica.ufmg.br/inetsec/uploadFiles/DOC/User_Manual.pdf

New tests will be required to find a balance of efficiency, accuracy and robustness. Hardware implemented tests are not new but are not implemented in many commercial RNGs that are not formally verified. We advise implementing such tests in any RNG claimed to be useful in a cryptography context. The PQ32MU, with its onboard post-processing and hardware tests of randomness, performs better on every test than the Quantis devices, demonstrating the benefits of this approach.

The key finding of this work is that current statistical testing of QRNGs relies on too few tests. Sample sizes frequently fall below the guidelines provided by the authors of statistical test batteries, even for RNGs that prove to be robust under far more stringent test conditions. We accept that RNG testing is not confined to statistical testing, entropy-source modelling and specific component tests for thermal-output and electro-magnetic leakage are also effective assessors of whether a RNG is fit for purpose. But statistical testing remains an accessible test method for consumers and independent testers, and a more diverse array of tests, with more stringent demands on data size, would be beneficial to manufacturers.

6 CONCLUSION

IDQ's Quantis generator exhibit particularly poor performance under TestU01 and Ent. Raw output not only fails TestU01 Alphabits, Rabbit and Crush but does so consistently on specific tests. This indicates an underlying problem, which although corrected by post-processing, calls into question the randomness of full-speed Quantis output. The issues identified in this work are exacerbated by the positive performance such generators exhibit under Dieharder or NIST SP800-22 tests. Positive results from these tests are used in marketing and certification documents, to state that these generators are suitable for use in cryptography when TestU01 and Ent imply otherwise. Reliance on a robust, but small, set of tests is an ongoing problem, which we have identified in previous work.

In the interests of responsible disclosure, we have contacted IDQ with our findings. Comscire's PQ32MU passed the statistical tests within expected parameters. As a result, we have not contacted Comscire at this time. Similarly, we have not identified any significant flaws in the ANU and Humboldt generators. All of the above entities have been contacted regarding our testing of their generators, but at time of writing only IDQ have responded.

Popular tests of randomness, in particular, those incorporated into the Dieharder and NIST SP800-22 batteries, are a useful means to identify egregious flaws in RNGs. However, this work has identified that these tests are often misused when testing is performed outside of official certification, leading to erroneous assertions regarding the quality of randomness that may be expected from a given device. Where certification procedures are followed, the tests are reliable, but may not identify flaws spotted by well-known but often overlooked test batteries (TestU01 and Ent demonstrating this in our work). We strongly recommend a wider selection of tests in both institution-backed and self-certification.

The extension of this work to Quantis AIS-31 certified products is a natural progression, as is obtaining a wider variety of QRNG devices for testing. By identifying manufacturing and post-processing techniques that appear to provide higher-quality randomness, future guidelines can be outlined for the statistical testing of randomness. Importantly, new tests are required to identify not only non-randomness but also the nature of non-randomness to allow for sophisticated diagnostic capabilities in the design, production and successive product iteration stages undertaken by the manufacturers of such devices. Hardware implementations of tests must be identified where possible, to improve the current state-of-the-art in online testing. This will form a core element of future work.

ACKNOWLEDGMENTS

This project has received funding from the European Union Horizon 2020 research and innovation programme, under grant agreement No.700326 (RAMSES project). This work has been partly funded by the EPSRC Quantum Communications Hub Project (EP/T001011/1). The authors would also like to thank EPSRC for project EP/P011772/1, on the EconoMical, PsycHologicAl and Societal Impact of RanSomware (EMPHASIS), which supported this work. The authors are thankful to the ECOST Cryptacus (ICT COST Action IC1403) project for the invaluable discussions and insights that have aided the development of this work, along with IDQ Semiconductors Ltd. for their timely and professional communication following the responsible disclosure of our findings.

REFERENCES

[1] JG Rarity, PCM Owens, and PR Tapster. Quantum random-number generation and key sharing. *Journal of Modern Optics*, 41(12):2435–2444, 1994.

[2] André Stefanov, Nicolas Gisin, Olivier Guinnard, Laurent Guinnard, and Hugo Zbinden. Optical quantum random number generator. *Journal of Modern Optics*, 47(4):595–598, 2000.

[3] ID Quantique. ID Quantique White Paper - Random Number Generation using Quantum Physics, April 2010.

[4] Darren Hurley-Smith and Julio Hernandez-Castro. Quam Bene Non Quantum: Identifying Bias in a Commercial Quantum Random Number Generator . Unpublished full-text manuscript from ResearchGate. Presented at Real World Crypto 2018, Zurich, Switzerland. http://bit.ly/2AOoiGF. Accessed: 2018-11-08.

[5] Daniel J Bernstein, Yun-An Chang, Chen-Mou Cheng, Li-Ping Chou, Nadia Heninger, Tanja Lange, and Nicko Van Someren. Factoring RSA keys from certified smart cards: Coppersmith in the wild. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 341–360. Springer, 2013.

[6] Xinzhe Wang, Futian Liang, Peng Miao, Yi Qian, and Ge Jin. 10-gbps true random number generator accomplished in asic. In *2016 IEEE-NPSS Real Time Conference (RT)*, pages 1–4. IEEE, 2016.

[7] Oto Petura, Ugo Mureddu, Nathalie Bochard, Viktor Fischer, and Lilian Bossuet. A survey of ais-20/31 compliant trng cores suitable for fpga devices. In *2016 26th international conference on field programmable logic and applications (FPL)*, pages 1–10. IEEE, 2016.

[8] Michal Varchola and Milos Drutarovsky. New high entropy element for fpga based true random number generators. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 351–365. Springer, 2010.

[9] Abdelkarim Cherkaoui, Viktor Fischer, Alain Aubert, and Laurent Fesquet. A self-timed ring based true random number generator. In *2013 IEEE 19th international symposium on asynchronous circuits and systems*, pages 99–106. IEEE, 2013.

[10] R Sivaraman, Sundararaman Rajagopalan, and Rengarajan Amirtharajan. Fpga based generic ro trng architecture for image confusion. *Multimedia Tools and Applications*, pages 1–28, 2020.

[11] IQ Quantique. *IDQ Random Number Generation*. IQ Quantique, http://www.idquantique.com/random-number-generation/, 2017.

[12] Bundesamt fur Sichterheit in der Informationstechnik. Evaluation of random number generators Version 0.10. Technical report, Bundesamt fur Sichterheit in der Informationstechnik, 2013.

[13] Krister Sune Jakobsson. Theory, methods and tools for statistical testing of pseudo and quantum random number generators, 2014.

[14] Pierre L'Ecuyer and Richard Simard. TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):22, 2007.

[15] Scott A Wilber. Entropy Analysis and System Design for Quantum Random Number Generators in CMOS Integrated Circuits. 2013.

[16] M Reznikov, R De Picciotto, M Heiblum, DC Glattli, A Kumar, and L Saminadayar. Quantum shot noise. *Superlattices and microstructures*, 23(3-4):901–915, 1998.

[17] Michael Wahl, Matthias Leifgen, Michael Berlin, Tino Röhlicke, Hans-Jürgen Rahn, and Oliver Benson. An ultrafast quantum random number generator with provably bounded output bias based on photon arrival time measurements. *Applied Physics Letters*, 98(17):171105, 2011.

[18] Thomas Symul, SM Assad, and Ping K Lam. Real time demonstration of high bitrate quantum random number generation with coherent laser light. *Applied Physics Letters*, 98(23):231103, 2011.

[19] Robert G Brown, Dirk Eddelbuettel, and David Bauer. Dieharder: A random number test suite. *Open Source software library, under development*, 2013.

[20] National Institute of Standards and Technology. *NIST SP800-22 Revision 1a A Statistical Test Suite for Random And Pseudorandom Number Generators for Cryptographic Applications*. Retrieved from:

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf 16:53 21/05/2018.

[21] George Marsaglia, Wai Wan Tsang, et al. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, 7(3):1–9, 2002.

[22] JY Haw, SM Assad, AM Lance, NHY Ng, V Sharma, PK Lam, and T Symul. Maximization of extractable randomness in a quantum random-number generator. *Physical Review Applied*, 3(5):054004, 2015.

[23] John Walker. *Ent. A pseudo-random number sequence testing program.* Retrieved from: https://www.fourmilab.ch/random/ 16:52 07/08/2018.

[24] Ziyong Zheng, Yichen Zhang, Weinan Huang, Song Yu, and Hong Guo. 6 gbps real-time optical quantum random number generator based on vacuum fluctuation. *Review of Scientific Instruments*, 90(4):043105, 2019.

[25] Saptadeep Pal, KK Soundra Pandian, and Kailash Chandra Ray. Fpga implementation of stream cipher using toeplitz hash function. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1834–1838. IEEE, 2014.

[26] Bingjie Xu, Ziyang Chen, Zhengyu Li, Jie Yang, Qi Su, Wei Huang, Yichen Zhang, and Hong Guo. High speed continuous variable source-independent quantum random number generation. *Quantum Science and Technology*, 4(2):025013, 2019.

[27] Kinga Marton and Alin Suciu. On the interpretation of results from the NIST statistical test suite. *Science and Technology*, 18(1):18–32, 2015.

[28] Walter Anderson. *A study of entropy.* Retrieved from: https://sites.google.com/site/astudyofentropy/background-information/the-tests 13:30 09/07/2018.

[29] Darren Hurley-Smith and Julio Hernandez-Castro. Certifiably Biased: An In-Depth Analysis of a Common Criteria EAL4+ Certified TRNG. *IEEE Transactions on Information Forensics and Security*, 13(4):1031–1041, 2018.

[30] Darren Hurley-Smith and Julio Hernandez-Castro. Bias in the Mifare DESFire EV1 TRNG. In *Radio Frequency Identification: 12th International Workshop, RFIDsec 2016, Hong Kong, China, November 30-December 2, 2016*. Springer International Publishing, 2016.

[31] Mario Rütti. A random number generator test suite for the c++ standard. *Institute for theoretical physics ETH Zurich, Diploma Thesis March*, 10, 2004.

[32] Pierre L'Ecuyer and Richard Simard. On the performance of birthday spacings tests with certain families of random number generators. *Mathematics and Computers in Simulation*, 55(1-3):131–137, 2001.

[33] George Marsaglia and Arif Zaman. Monkey tests for random number generators. *Computers & mathematics with applications*, 26(9):1–10, 1993.

[34] Wolfgang Killmann and Werner Schindler. A proposal for: Functionality classes for random number generators. *ser. BDI, Bonn*, 2011.

[35] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry McKay, Mary Baish, and Michael Boyle. Recommendation for the entropy sources used for random bit generation. Technical report, National Institute of Standards and Technology, 2016.

[36] Shuangyi Zhu, Yuan Ma, Tianyu Chen, Jingqiang Lin, and Jiwu Jing. Analysis and improvement of entropy estimators in NIST SP 800-90B for Non-IID entropy sources. *IACR Transactions on Symmetric Cryptology*, pages 151–168, 2017.

[37] Pierre L'Ecuyer, Richard Simard, and Stefan Wegenkittl. Sparse serial tests of uniformity for random number generators. *SIAM Journal on scientific computing*, 24(2):652–668, 2002.

[38] Damian Twerendol and Philippe Richard. Certificate of Conformity No 151-04687, May 2010.

[39] Compliance Testing Laboratory. Certificate of Compliance, March 2011.

7 APPENDIX: TEST BATTERY CONFIGURATION

The Randomness Testing Toolkit (RTT)[20] was used to conduct the Dieharder, NIST SP 800-22 and TestU01 experiments. Ent was used as provided in the Ubuntu 16.04 LTS distribution. Modified parameters are discussed below.

7.1 Dieharder

Battery parameters required some modification due to their small sample size. The following tests were used for all files great than 1GB in size:

- Birthdays
- OPERM5
- 32x32 Binary Rank
- 6x8 Binary Rank
- Count the 1's (stream)
- Count the 1's (byte)
- Parking Lot
- Minimum Distance (2d Circle)
- 3d Sphere (Minimum Distance) test
- Squeeze
- Runs
- Craps
- Marsaglia and Tsang GCD
- STS Monobit
- STS Runs
- STS Serial (Generalized)
- RBG Bit Distribution
- RGB Generalized Minimum Distance
- RGB Permuations
- RGB Lagged Sum
- Kolgorov-Smirnov (excluded for ANU)
- DCT (Frequenct Analysis)
- DAB Fill Tree (excluded for ANU)
- DAB Fill Tree 2 (excluded for ANU)
- DAB Monobit 2 (excluded for ANU)

Four fewer tests (23 instead of 27) are conducted over ANU sequences (noted in the list). Tests that rewind during a single run are omitted. Rewinds are tolerated so long as the sequence does not reach the end of the file during a single test.

7.2 TestU01 - Crush

Crush has been executed using a reduced set of tests to allow for testing 2.1GB files (test numbers provided in brackets next to each test):

- smarsa CollisionOver (3-10)
- smarsa BirthdaySpacings (11)
- snpair ClosePairs (18-20)
- snpair ClosePairsBitMatch (21 and 22)
- sknuth Max0ft (43)

[20] https://github.com/crocs-muni/randomness-testing-toolkit

- svaria SampleProd (45)
- svaria SampleMean (47)
- svaria AppearanceSpacings (49 and 50)
- smarsa MatrixRank (56)
- smarsa MatrixRank (58)
- smarsa MatrixRank (60)
- smarsa GCD (63-64)
- swalk RandomWalk1 (65-70)
- scomp LinearComp (71-72)
- scomp LempelZiv (73)
- sspectral Fourier3 (74 and 75)
- sstring HammingIndep (90)
- sstring Run (91)

Big Crush has been omitted, as it requires substantially more data than was included in the sample files.